Handle All Errors Safely

William L. Fithen, Software Engineering Institute [vita[3]]

2005-10-03 L4 / D/P[4]

Unhandled or incorrectly handled exceptions can introduce vulnerability.

Description

Error handling (or, more generally, "exception handling") is an essential aspect of reliability and security and is frequently a source of vulnerabilities. An "error" is an internal state not expected to occur during normal operation that, if not handled correctly, could lead to a failure to deliver the required service. Error handling consists of detecting such internal states and performing the required reaction to such events. The reaction might be some form of recovery such as a retry of a failed calculation.

In the case of errors that cannot be corrected, the Failing Securely[8] principle councils one to take the most conservative reaction. The required reaction may consist of a "graceful" exit with appropriate notification of the failure to the calling procedure or program and to any components monitoring system health. Frequently the called function's return value is used to indicate an error, but a failure might also raise an exception, invoke a callback, or schedule a signal.

Successful error management also depends on the calling procedure recognizing and correctly reacting to an error generated on a call. However, the calling procedure itself might not have sufficient context to be able to handle the error in the way that is best for the system as a whole, especially in an object oriented systems. Therefore, the calling procedure may pass the exception up to its own calling procedure (perhaps with some additional context information) to improve the chances for successful recovery from an error. At the highest level, every error must be dealt with appropriately or extreme measure are required. The typical handler for unexpected top level exceptions is to terminate. But other reactions are possible, such as complete reinitialization of the system.

An ignored error or an improper reaction to an error will likely generate an inconsistent or unanticipated internal state in the calling program. "Unanticipated" implies that neither detection nor recovery is available for such errors and that a vulnerability, whose exploitation can lead to a system failure or an insecure state, may be possible [VU#795632[9]].

The mechanisms used to report errors can also induce vulnerabilities. In languages that support exceptions, the evolution of a component may encourage the interception and silent suppression of new classes of exceptions within the component in order to preserve the component's original API, since raising or propagating new exceptions may require syntactical or semantic changes in calling code. The suppression of such exceptions can produce unexpected results, which can lead to vulnerabilities of this class. The prototypical case is the suppression of "checked" exceptions in Java.

Since error handling code typically deals with rare or infrequent events, it is not unusual for insufficient effort to be devoted to developing such code and for thorough testing of error handling code to be overlooked. This contributes to a lessening of reliability for the system as a whole and an increased probability that a vulnerability related to error handling exists.

3. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/320-BSI.html (Fithen, William L.)
8. 349.html
9. Contrary to what this looks like, this is really a failure to catch an error, not a buffer overflow.

References

[VU-238678]	Gennari, Jeff. *Vulnerability Note VU#238678: The zlib compression library is vulnerable to a denial-of-service condition*. http://www.kb.cert.org/vuls/id/238678 (2004).
[VU#331937]	Finlay, Ian. *Vulnerability Note VU#331937: BEA WebLogic Server "ResourceAllocationException" exception may disclose user password*. http://www.kb.cert.org/vuls/id/331937 (2003).
[VU#795632]	Manion, Art. *Vulnerability Note VU#795632: MIT Kerberos 5 ASN.1 decoding functions insecurely deallocate memory (double-free)*. http://www.kb.cert.org/vuls/id/795632 (2005).

Carnegie Mellon Copyright

1. mailto:permission@sei.cmu.edu